DEPARTMENT OF APPLIED ECONOMICS

RESEARCH REPORT

AN APPROXIMATION ALGORITHM FOR A GENERALIZED ASSIGNMENT PROBLEM WITH SMALL RESOURCE REQUIREMENTS

JOEP AERTS • JAN KORST • FRITS C.R. SPIEKSMA

OR 0363

An approximation algorithm for a generalized assignment problem with small resource requirements*

Joep Aerts†, Jan Korst‡ and Frits C.R. Spieksma§

October 8, 2003

Abstract

We investigate a generalized assignment problem where the resource requirements are either 1 or 2. This problem is motivated by a question that arises when data blocks are to be retrieved from parallel disks as efficiently as possible. The resulting problem is to assign jobs to machines with a given capacity, where each job takes either one or two units of machine capacity, and must satisfy certain assignment restrictions, such that total weight of the assigned jobs is maximized. We derive a $\frac{2}{3}$-approximation result for this problem based on relaxing a formulation of the problem so that the resulting constraint matrix is totally unimodular. Further, we prove that the LP-relaxation of a special case of the problem is half-integral, and we derive a weak persistency property.

Key words: parallel disks, generalized assignment problem, retrieval problem, half-integrality, persistency

1 Introduction

We consider the following problem. Given is a set of jobs $J = \{1, 2, \ldots, n\}$ and a set of machines $M = \{1, 2, \ldots, m\}$. Each machine has capacity b_i, $i \in M$. Each job j can be assigned to a job-specific subset of the machines denoted by $M(j) \subseteq M$. Each $M(j)$ is partitioned into two subsets, $M_1(j)$ and $M_2(j)$, such that assigning job j to machine i in $M_1(j)$ takes one unit of capacity and to a machine in $M_2(j)$ two units. Assigning job j to machine i results in a given profit of w_{ij}. The problem is to assign jobs to machines such that total

*A preliminary version of this work appeared as [3].

†Centre of Quantitative Methods, P.O. Box 414, NL-5600 AK Eindhoven, The Netherlands. aerts@cqm.nl

‡Philips Research Labs, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands. jan.korst@philips.com

§Department of Applied Economics, Katholieke Universiteit Leuven, Naamsestraat 69, B-3000, Leuven, Belgium. frits.spieksma@econ.kuleuven.ac.be

weight of assigned jobs is maximized while respecting the capacities and the assignment restrictions induced by the sets $M(j)$. We will refer to this problem as problem P. A straightforward formulation of P is as follows. For each $j \in J$ and $i \in M(j)$ we introduce a decision variable

$$x_{ji} = \begin{cases} 1 & \text{if job } j \text{ is assigned to machine } i \\ 0 & \text{otherwise.} \end{cases}$$

The model now is:
(M1) Maximize

$$\sum_{j \in J} \sum_{i \in M(j)} w_{ji} x_{ji}$$

subject to:

$$\forall_{j \in J} \qquad \sum_{i \in M(j)} x_{ji} \leq 1, \tag{1}$$

$$\forall_{i \in M} \qquad \sum_{j \in J_1(i)} x_{ji} + 2 \sum_{j \in J_2(i)} x_{ji} \leq b_i, \tag{2}$$

$$\forall_{j \in J} \; \forall_{i \in M(j)} \qquad x_{ji} \in \{0, 1\}, \tag{3}$$

with $J_1(i)$ and $J_2(i)$ being the sets of jobs that take one, respectively two, units of capacity when assigned to machine i. Constraints (1) express that each job can be assigned at most once, (2) state that the capacity of each machine should not be violated, and (3) are the integrality constraints.

Problem P is intimately related to the so-called Generalized Assignment Problem (GAP). In the literature (see Section 1.1) different variations of formulations of the GAP exist: maximization versions versus minimization versions, and equality constraints versus the current inequality constraints (1). Suggested by the application that motivated our work (see Section 2), we have opted in our formulation for maximization under inequality constraints; notice however that the results also apply in case equality constraints are used, assuming a feasible solution exists. Of course, a crucial property of problem P is that the resource requirements, usually denoted by a_{ji}, are equal to 1 or 2: in that sense P is a special case of GAP. Another property of P is that it takes explicitly into account, using the sets $M(j)$, that a job cannot go to an arbitrary machine: in that sense P is more general than GAP.

A special case of problem P arises when the resource requirement of a job j is independent of the machine i, i.e., $a_{ji} = a_j$. In this case, we refer to the resource requirement of job j as its *size*. Thus, in such a setting, job j either has size 1 on all machines in $M(j)$, i.e., $M(j) = M_1(j)$, or it has size 2 on all machines in $M(j)$, i.e., $M(j) = M_2(j)$. The resulting problem, denoted by P12, is a special case of the so-called multiple knapsack problem with assignment restrictions, as in our setting the size of each job is 1 or 2. This special case is motivated in Section 2.

In the next section we describe related literature, our results and the setup of this paper.

1.1 Related literature and our results

The generalized assignment problem is a basic problem in operations research. An exact algorithm has been described by Savelsbergh [18]. Polyhedral aspects of the GAP have been studied as well, see De Farias and Nemhauser [9] for a recent reference. Often, the GAP is stated as a minimization problem with the restriction that each job has to be assigned; by scaling the resource requirements one may then assume that in the GAP all machine capacities are equal, i.e., $b_1 = b_2 = \ldots = b_m = b$. A seminal work on approximating the GAP is Shmoys and Tardos [19]. Given a value C and a value b, they show how to compute in polynomial time an assignment with cost at most C (if such an assignment exists), needing machine capacities at most $2b$.

As far as we are aware, the special cases described here, i.e., P and P12, have not been investigated before. We describe in Section 2 our motivation for investigating this special case. In Section 3 we state the complexity of our problem and provide an integer programming formulation of P as an alternative to M1. We show how this formulation can be relaxed such that the resulting constraint matrix is totally unimodular (TUM). Based on this relaxation we sketch a $\frac{2}{3}$-approximation algorithm whose complexity is determined by solving a transportation problem. Another setting where an integer program is transformed to an integer program with a TUM constraint matrix appears is described in Hochbaum [11]. In particular, her framework focuses on formulations where there is a unique variable for each constraint in the formulation.

In Section 4 we investigate P12. A problem related to P12 is discussed by Dawande et al. [8]: their setting differs from P12 in the sense that the size of a job a_j is not restricted to $\{1, 2\}$; they assume however, that the profit of a job equals its size, i.e., $w_{ji} = a_j$ for all $j \in J, i \in M$. They describe two approaches that each lead to a $\frac{1}{2}$-approximation algorithm. For P12 we show that the LP-relaxation and the TUM-relaxation coincide. Moreover, it turns out that the LP-relaxation of M1 for this special case is half-integral. This property has been studied by Hochbaum [11] and Appa and Kotnyek [6]. Hochbaum describes a technique that, based on first finding a half-integral superoptimal solution, delivers 2-approximations for a large class of minimization problems. Appa and Kotnyek [6] investigate a class of matrices for which the associated polytope has half-integral vertices only. Other occurrences of half-integrality are described in Chudak and Hochbaum [7] and Ralphs [14]. We also show that in the case of unit weights, i.e., $w_{ji} = 1$ for all $j \in J, i \in M$, a weak persistency property holds. More specifically, it turns out that there exists an optimal solution to P12 such that jobs of size 1 not assigned in the LP-relaxation are also not assigned in that optimal solution and vice versa.

2 The application

Storage and retrieval problems for parallel disks have been studied extensively in the area of video on demand, see e.g. Aerts [2], and external memory al-

gorithms, see e.g. Vitter [20]. When handling large collections of data, the communication between fast internal memory and slow external memory, e.g. disks, can be a major performance bottleneck. Several disk storage strategies have been proposed that use redundancy to balance the load on parallel disks, see e.g. Aerts, Korst and Verhaegh [5] or Sanders, Egner and Korst [17]. Storing multiple copies of each data block on different disks allows one to dynamically determine from which disk to retrieve a requested block. This improves the throughput, i.e., the number of requests served per time unit.

The retrieval of data from parallel disks can be modeled as follows. Given is a set of identical disks on which equal-sized blocks are stored redundantly. Requests for blocks arrive over time and repeatedly a batch of blocks is retrieved. Under the assumption that a disk can only retrieve one block per cycle, the problem of maximizing the number of blocks retrieved in a cycle can be modeled as a maximum flow problem [2].

In reality, however, this assumption is not fulfilled. In fact, the disks can be used more efficiently if we exploit the multi-zone character of hard disks (see Ruemmler and Wilkes [15]). Hard disks rotate at a constant angular velocity, while outer tracks contain more data than inner tracks. Hence, one can retrieve data at a higher rate from the outside than form the inside. One can partition a disk's storage space into two halves, an inner half and an outer half. Given this partition, it is reasonable to assume that the worst-case retrieval rate at the outer half is approximately twice the worst-case retrieval rate achievable at the inner half.

Using the above, we consider the following improvement. Instead of retrieving at most one block per disk per cycle, each disk can now either retrieve two blocks from its outer half or one block from its inner half. In this way, we can considerably increase the average number of blocks read per cycle, with little or no increase in the average cycle duration. Given that blocks are stored redundantly, the problem of maximizing the number of retrieved blocks is no longer polynomially solvable, provided $\mathcal{P} \neq \mathcal{NP}$.

Alternative ways to improve the throughput of parallel disks are (*i*) further increasing the batch sizes to decrease seek overhead (see e.g. Gemmell [10]), and (*ii*) handling the disks in an asynchronous fashion (see e.g. Sanders [16]). These alternatives fall outside the scope of this paper.

If we refer to block requests as jobs and to disks as machines, the retrieval problem can be considered as nonpreemptively assigning jobs to machines, with jobs having a size one or two and machines having capacity two, where each job can only be assigned to a subset of the machines. The objective is to maximize the number of assigned jobs. This gives rise to instances of problem P with unit weights, and $b_i = 2$ for all $i \in M$. Notice that although this may look easier than problem P itself, Theorem 1 implies that this special case of P is already difficult to approximate arbitrary closely in polynomial time.

3 Approximation of P

Recall problem P as introduced in Section 1: given is a set J of jobs, and a set $M = \{1, \ldots, m\}$ of machines, with machine $i \in M$ having capacity b_i. Each job j can only be assigned to a machine i from a given set $M(j)$. Moreover, for each machine i in $M(j)$ it is known whether job j takes one unit of capacity or two units of capacity. Finally, there is a given weight w_{ij} for assigning job j to machine i. All data are positive integral numbers. We assume that no machine with capacity 1 is in $M_2(j)$ for some $j \in M$. This assumption is quite reasonable since in no feasible solution a machine with capacity 1 can process a size 2 job. The problem is to find an assignment of jobs to machines that maximizes the weighted number of jobs that are assigned.

Let us first phrase a result from Aerts et al. [4] in terms of our problem P.

Theorem 1 (Aerts et al. [4]) *P12 is APX-hard, even if each machine has capacity 2, each job is connected to exactly 2 machines, and all weights are 1, i.e., if $b_i = 2$ for all $i \in M$ and $|M(j)| = 2$ and $w_j = 1$ for all $j \in J$.*

Notice that this result for P12 applies a fortiori to problem P.

3.1 A formulation of P and its relaxations

We now present an alternative model for problem P. This alternative formulation allows us to investigate two relaxations of the formulation. Instead of jobs that need either one or two units of capacity of a particular machine, we introduce for each job j a regular job $r(j)$ and a dummy job $d(j)$. A regular job, as well as a dummy job has size 1. Job $r(j)$ is connected to each machine i that is in $M(j)$, i.e., $M(r(j)) = M(j)$, whereas job $d(j)$ is connected to each machine i that is in $M_2(j)$, i.e., $M(d(j)) = M_2(j)$. In case $M_2(j) = \emptyset$, we do not create job $d(j)$. We choose the weights for the jobs $r(j)$ and $d(j)$ as follows: the weight for assigning job $r(j)$ to a machine i in $M_1(j)$ is w_{ij}; the weight for assigning job $r(j)$ to a machine i in $M_2(j)$ is $\frac{1}{2}w_{ij}$; and the weight for assigning job $d(j)$ to a machine i in $M_2(j)$ is $\frac{1}{2}w_{ij}$. This gives the following, alternative, formulation for P.

(M2) Maximize

$$\sum_{j \in J} \sum_{i \in M_1(j)} w_{ji} x_{r(j),i} + \frac{1}{2} \sum_{j \in J} \sum_{i \in M_2(j)} w_{ji} (x_{r(j),i} + x_{d(j),i})$$

subject to:

$$\begin{aligned}
\forall_{j \in J} \quad & \sum_{i \in M(j)} x_{r(j),i} \leq 1, \\
\forall_{j \in J} \quad & \sum_{i \in M_2(j)} x_{d(j),i} \leq 1, \\
\forall_{i \in M} \quad & \sum_{j \in J_r(i)} x_{r(j),i} + \sum_{j \in J_d(i)} x_{d(j),i} \leq b_i,
\end{aligned}$$

$$\forall_{j\in J}\ \forall_{i\in M_2(j)} \qquad x_{r(j),i} - x_{d(j),i} = 0, \tag{4}$$

$$\forall_{j\in J}\ \forall_{i\in M(j)} \qquad x_{r(j),i}, x_{d(j),i} \in \{0,1\}, \tag{5}$$

with $J_r(i)$ and $J_d(i)$ being the sets of regular and dummy jobs that can be assigned to machine i respectively.

Note that constraints (4) and (5) imply that either both jobs $r(j)$ and $d(j)$ are assigned to machine $i \in M_2(j)$ or both jobs are not assigned to machine i. Thus, model M2 is a valid formulation of P.

A traditional way of finding a solution to an integer programming model is to first solve the LP-relaxation, and next apply a rounding procedure. Here, we opt for a different strategy: we investigate the relaxation of model M2 induced by deleting constraints (4). We refer to this relaxed problem as the TUM-relaxation. There are two reasons that motivate the choice to drop the constraints (4). First of all, the resulting model has a constraint matrix that is totally unimodular. Even more, we argue below that this relaxation can be computed by solving a transportation problem on a network consisting of $O(n+m)$ nodes. This is not so obvious for the LP-relaxation; indeed we do not know how to compute the LP-relaxation of a generalized assignment problem using min-cost flow even when the resource requirements are in $\{1,2\}$. Secondly, although we show below that the bound resulting from the LP-relaxation is stronger than the bound resulting from the TUM-relaxation, the integrality gap of the LP-relaxation suggests that an algorithm based on the LP-relaxation will not improve the $\frac{2}{3}$-performance guarantee that arises from the TUM-relaxation. Summarizing, the TUM-relaxation seems easier to compute than the LP-relaxation, and does not deliver worse results in terms of approximation factor.

We use the following terminology to describe our results.

- Let LPM1 and LPM2 denote the linear programming relaxation of formulation M1 and M2, respectively, i.e., the model that results when replacing constraints (3) or (5) by the nonnegativity constraints.
- As stated before we refer to the problem that arises when omitting constraints (4) from formulation M2 as the *TUM-relaxation* of M2.
- Let $\mathcal{I}$ denote an instance of P, and let $v_{LPM1}(\mathcal{I}), v_{LPM2}(\mathcal{I}), v_{TUM}(\mathcal{I})$ and $OPT(\mathcal{I})$ denote the corresponding values of the respective relaxations and the optimal solution. For notational convenience, we will often suppress the '$(\mathcal{I})$' part.
- Let x^{TUM} denote an optimal solution to the TUM-relaxation, and let x^{LP} denote an optimal solution to the LP-relaxation of M2.

Let us first establish that the LP-relaxations of models M1 and M2 coincide.

Lemma 2 *$v_{LPM1}(\mathcal{I}) = v_{LPM2}(\mathcal{I})$ for all instances $\mathcal{I}$.*

Proof: Take a solution, say y, that is feasible for the LP-relaxation of M1. We set

- for all $j \in J, i \in M_1(j)$: $x_{r(j),i} = y_{ji}$,
- for all $j \in J, i \in M_2(j)$: $x_{r(j),i} = y_{ji}$ and $x_{d(j),i} = y_{ji}$.

Notice that the solution constructed is feasible for LPM2 (by the feasibility of y for LPM1) and has the same value.

For the converse, assume that we are given a solution y that is feasible for LPM2. We simply set: $x_{ji} = y_{r(j),i}$ for all $j \in J, i \in M(j)$. It is not hard to verify that the resulting x-solution is feasible for LPM1 and has the same value. □

In the sequel, we use v_{LP} to refer to v_{LPM1} and v_{LPM2}. Let us now explain how the TUM-relaxation of M2 can be computed efficiently.

Theorem 3 *v_{TUM} can be computed by solving a transportation problem.*

Proof: Let the network of a transportation problem be denoted by $G = (V_1 \cup V_2, A)$. For each job $r(j)$ and for each job $d(j)$ there is a so-called job-node in V_1. Also, there is a single additional node $d \in V_1$. The supply of each job-node equals 1, the supply of node d equals $\sum_{i=1}^{m} b_i$. For each machine i, there is a so-called machine node in V_2, and V_2 also contains a node e. For each machine i, the demand of the corresponding machine-node equals b_i and the demand of node e equals $|V_1| - 1$. Notice that total supply equals total demand. There is an arc from each job-node $r(j)$ to each machine-node $i \in M_1(j)$ with cost $-w_{ji}$; there is an arc from each job-node $r(j)$ to each machine-node $i \in M_2(j)$ with cost $-\frac{1}{2}w_{ji}$; there is an arc from each job-node $d(j)$ to each machine-node $i \in M_2(j)$ with cost $-\frac{1}{2}w_{ji}$. Finally, there are arcs from node d to each machine-node i with cost 0, and there is an arc from each job-node to node e with cost 0, and there is an arc (d, e) with cost 0. This completes the description of the network. The objective of the transportation problem is to minimize cost.

We now argue that a feasible flow in this network corresponds to a feasible solution to the TUM-relaxation of M2 with the same value and vice versa. Let x be a solution to the TUM-relaxation of M2. We associate the following flow with this solution. Each variable $x_{r(j),i}$ and $x_{d(j),i}$ corresponds directly to an arc between a job-node and a machine-node. We set the flow on this arc equal to the x-variable. Given these flows, we use the arcs that go to node e and that emanate from node d to satisfy the supply and demand conditions. This is always possible since each job-node is connected to e, and d is connected to each machine node. Observe that the value of the resulting flow equals the value of the solution x.

Consider now any feasible flow. Since each arc between a job-node and a machine-node corresponds to a variable in the TUM-relaxation of M2, we simply set the value of this variable equal to the flow on this arc. Notice that the constructed x-vector satisfies all constraints in M2 and has the same value. Observe finally that, due to the integrality of an optimal solution to the transportation problem, a solution of the TUM-relaxation of M2 exists that satisfies the integrality constraints (5). □

Corollary 4 *v_{TUM} can be computed in $O(nmlogn + n^2log^2n)$.*

This time bound follows from Orlin [13].

Lemma 5 *$v_{LP} \leq v_{TUM}$.*

Proof: We show that a solution that is feasible for the LP-relaxation can be seen as a flow in the network described in Theorem 3. Then it follows that v_{TUM} cannot be less than v_{LP}. Consider an LP-solution of M2. Simply copy the value of each $x_{r(j),i}$ and $x_{d(j),i}$ variable as a flow in this network on the appropriate arc. Observe that the value of this flow equals the value of the LP-solution. □

The following example shows that the LP-relaxation can produce a strictly better upper bound than the TUM-relaxation. Consider two machines with $b_1 = b_2 = 2$ and two jobs, job 1 with $a_{11} = 1, a_{12} = 2$, and job 2 with $a_{21} = 2, a_{22} = 1$. We have $w_{ji} = 1$ for all $j \in J, i \in M$. One can easily verify that $OPT = v_{LP} = 2$, whereas $v_{TUM} = 3$. Figure 1 depicts the solution of the TUM-relaxation.

Figure 1: Example that shows that TUM-relaxation can be worse than LP-relaxation.

Notice however that the gap between v_{TUM} and v_{LP} cannot exceed $\frac{3}{2}$. This follows from Theorem 7.

Let us now formulate a lemma that reveals some of the structure that is present in x^{TUM}. Informally, it shows that x^{TUM} is not only an optimal solution for instance $\mathcal{I}$, but also for an instance $\mathcal{I}'$ that consists of a subset of the jobs of $\mathcal{I}$ and uses machines with a specific fraction of the capacity of the machines in $\mathcal{I}$. An instance of problem P can be described by its job set J, the machine subsets $M_1(j)$ and $M_2(j)$, $j \in J$, the machine capacities $b_i, i \in M$, and the weights w_{ji}. Succinctly put: $\mathcal{I} = (J, \{M_p(j) |\ j \in J, p = 1, 2\}, b, w)$.

To build a new instance $\mathcal{I}'$, we consider now a set of jobs $J' \subseteq J$, and we define, for each $i \in M$, the capacity of machine i as:

$$b'_i = \sum_{j \in J'} x^{TUM}_{ji}.$$

Thus the capacity of machine i equals the number of jobs in J' assigned to machine i in the TUM-relaxation of instance $\mathcal{I}$. Let w' refer to the weight vector w restricted to jobset J'.

Lemma 6 *For any $J' \subseteq J$ it is true that $x^{TUM}_{ji}, j \in J', i \in M$ is an optimal solution for the TUM-relaxation of $\mathcal{I}' = (J', \{M_p(j) |\ j \in J', p = 1, 2\}, b', w')$.*

Proof: Obviously $x^{TUM}_{ji}, j \in J', i \in M$ is a feasible solution for the TUM-relaxation of $\mathcal{I}'$ since it does not exceed the capacities b' by its construction.

Also, if a solution y to the TUM-relaxation of $\mathcal{I}'$ existed with a higher value, this would contradict the optimality of x^{TUM} for $\mathcal{I}$: indeed the solution $x^{TUM}_{ji}, j \in J \setminus J', i \in M$, together with y would be feasible to the TUM-relaxation of $\mathcal{I}$ and would attain a higher value. □

We need Lemma 6 in the proof of the main result of this section:

Theorem 7 $v_{TUM} \leq \frac{3}{2} OPT$.

Proof: The idea of the proof is to show that a solution to the TUM-relaxation of M2 can be modified into a feasible solution to M2 without losing too much weight. Our approach will consist in constructing three feasible solutions to M2, and we argue that the best of these three solutions has a weight of at least $\frac{2}{3}$ times the weight of the solution to the TUM-relaxation of M2.
Definition: A job j is called *split* when

- there exist machines $i_1 \in M_1(j)$ and $i_2 \in M_2(j)$ such that $x^{TUM}_{r(j),i_1} = x^{TUM}_{d(j),i_2} = 1$, in case we say that job j is of type 1, or
- there exist machines $i_1 \in M_2(j)$ and $i_2 \in M_2(j)$, $i_1 \neq i_2$ such that $x^{TUM}_{r(j),i_1} = x^{TUM}_{d(j),i_2} = 1$, than we say that job j is of type 2, or
- there exists a machine $i_1 \in M_2(j)$ such that $x^{TUM}_{r(j),i_1} = 1$, which makes job j of type 3.

We assume that the solution of the TUM-relaxation has the property that if $d(j)$ is assigned, then $r(j)$ is assigned as well. Notice that a solution violating this property is easily modified into a solution satisfying this property by interchanging $r(j)$ and $d(j)$.

Also notice that a job j that is not split satisfies constraints (4); hence, a solution x^{TUM} featuring no split jobs is a feasible and necessarily optimal solution to the instance of problem P.
Definition: A machine i is called *fractional* when there exists a split job j for which $x^{TUM}_{r(j),i} = 1$ or $x^{TUM}_{d(j),i} = 1$, $i \in M$.
Definition: A machine i has a *free unit* with respect to x when $\sum_{j \in J}(x_{r(j),i} + x_{d(j),i}) \leq b_i - 1$, $i \in M$.

Consider now the following claim:

Claim: There exist three feasible solutions to M2, denoted by S_1, S_2, S_3 with values $v(S_1), v(S_2), v(S_3)$ respectively, such that

- $\max\{v(S_1), v(S_2), v(S_3)\} \geq \frac{2}{3} v_{TUM}$,
- each fractional machine has a free unit in at least one solution from $\{S_1, S_2, S_3\}$, and
- if all split jobs are of type 2, each fractional machine has a free unit in at least two solutions from $\{S_1, S_2, S_3\}$.

To prove the claim, we use induction on the number of split jobs. Let k be the number of split jobs. Consider the case $k = 1$, and let us refer to the split job as job s. We use the following definition.
Definition: For each machine $i \in M$, we define

$$W(i) = \sum_{j \in J_1(i) \setminus s} w_{ji} x^{TUM}_{r(j),i} + \sum_{j \in J_2(i) \setminus s} \frac{1}{2} w_{ji} (x^{TUM}_{r(j),i} + x^{TUM}_{d(j),i}).$$

Thus $W(i)$ represents the weight of the jobs assigned to machine i in the TUM-relaxation, excluding job s, $i \in M$. We distinguish three cases. For each of the three cases, we construct three, not necessarily different, solutions S_1, S_2, and S_3, that satisfy the above claim.

Case 1: The split job, i.e., job s, is of type 1. We now construct the three solutions S_1, S_2, S_3. Obviously, for each machine $i \notin \{i_1, i_2\}$ we simply copy the assignment of jobs assigned to machine i in the TUM-relaxation to each of the three solutions. Let us now deal with machines i_1 and i_2. Consider S_1: for machines i_1 and i_2, we copy the assignment of each job to the machine as suggested by the TUM-relaxation, except that we assign job s to machine i_1. Solution S_2 is identical to solution S_1. We have:

$$v(S_1) = v(S_2) = \sum_{i=1}^{m} W(i) + w_{s,i_1}.$$

Notice that S_1 and S_2 are feasible solutions since job s takes one unit on machine i_1. For S_3, we again copy the assignment of jobs to machine i_1 as suggested by the TUM-relaxation, except that we do not assign job s to machine i_1. Consider now machine i_2. It must have capacity at least 2, i.e., $b_{i_2} \geq 2$, by assumption. Thus, there are two possibilities: i) there is a free unit of capacity in the TUM-relaxation. Then we simply put job s on machine i_2, and copy all other assignments. ii) there is no free unit. That means there is another (non-split) job present on machine i_2, say r. We assign job s to machine i_2 and do not assign job r to machine i_2, and copy all other assignments. In either case we have:

$$v(S_3) \geq \sum_{i=1}^{m} W(i) + w_{s,i_2} - w_{r,i_2}.$$

Let us now verify whether the claim is true. First, observe that

$$\max\{v(S_1), v(S_2), v(S_3)\} \geq \frac{1}{3}[v(S_1) + v(S_2) + v(S_3)] =$$
$$\frac{1}{3}[3\sum_{i=1}^{m} W(i) + 2w_{s,i_1} - w_{r,i_2} + w_{s,i_2}] =$$
$$\sum_{i=1}^{m} W(i) + \frac{2}{3} w_{s,i_1} - \frac{1}{3} w_{r,i_2} + \frac{1}{3} w_{s,i_2} \geq$$

$$\frac{2}{3}[\sum_{i=1}^{m} W(i) + w_{s,i_1} + \frac{1}{2}w_{s,i_2}] = \frac{2}{3}v_{TUM}.$$

Notice that the term $\sum_{i=1}^{m} W(i)$ contains w_{r,i_2}.

Second, since machine i_1 has a free unit with respect to S_3, and machine i_2 has a free unit with respect to S_1 and S_3, it follows that each fractional machine has a free unit in at least one solution from S_1, S_2, S_3. Thus the claim is valid in case job s is of type 1.

Case 2: Job s is of type 2. We now construct the three solutions S_1, S_2, S_3. Obviously, for each machine $i \notin \{i_1, i_2\}$ we simply copy the assignment of jobs assigned to machine i in the TUM-relaxation to each of the three solutions. Let us now deal with machines i_1 and i_2. Observe that we have $b_{i_1} \geq 2$ and $b_{i_2} \geq 2$. For solution S_1, consider first machine i_1. There are two possibilities: i) there is a free unit. Then we simply put job s on machine i_1, and copy all other assignments of jobs to machines i_1 and i_2 in the TUM-relaxation. ii) there is no free unit. That means there is another (non-split) job present on machine i_1, say r. We assign job s to machine i_1 and do not assign job r to machine i_1, while copying all other assignments. We have:

$$v(S_1) \geq \sum_{i=1}^{m} W(i) + w_{s,i_1} - w_{r,i_1}.$$

A similar strategy (leading to the assignment of job s on machine i_2) is employed to construct S_2, possibly deleting a non-split job p, arriving at

$$v(S_2) \geq \sum_{i=1}^{m} W(i) + w_{s,i_2} - w_{p,i_2}.$$

S_3 is constructed by not assigning job s, while copying all the assignments of other jobs to machines:

$$v(S_3) = \sum_{i=1}^{m} W(i).$$

Let us now verify whether the claim is true. First, observe that

$$\max\{v(S_1), v(S_2), v(S_3)\} \geq \frac{1}{3}[v(S_1) + v(S_2) + v(S_3)] =$$

$$\frac{1}{3}[3\sum_{i=1}^{m} W(i) + w_{s,i_1} + w_{s,i_2} - w_{r,i_1} - w_{p,i_2}] =$$

$$\sum_{i=1}^{m} W(i) + \frac{1}{3}w_{s,i_1} + \frac{1}{3}w_{s,i_2} - \frac{1}{3}w_{r,i_1} - \frac{1}{3}w_{p,i_2} \geq$$

$$\frac{2}{3}[\sum_{i=1}^{m} W(i) + \frac{1}{2}w_{s,i_1} + \frac{1}{2}w_{s,i_2}] = \frac{2}{3}v_{TUM}.$$

Notice that the term $\sum_{i=1}^{m} W(i)$ contains w_{r,i_1} and w_{p,i_2}.

Second, since machine i_1 has a free unit with respect to S_2 and S_3, and machine i_2 has a free unit with respect to S_1 and S_3, it follows that each fractional machine has a free unit in at least two solutions from S_1, S_2, S_3. Thus the claim is valid in case job s is of type 2.

Case 3: Job s is of type 3. We now construct the three solutions S_1, S_2, S_3. Obviously, for each machine $i \in M \setminus i_1$ we simply copy the assignment of jobs assigned to machine i in the TUM-relaxation to each of the three solutions. Let us now deal with machine i_1. Observe that we have $b_{i_1} \geq 2$. Let us first construct S_1. There are two possibilities for machine i_1: i) there is a free unit. Then we simply put job s on machine i_1, and copy all other assignments of jobs to machine i_1 in the TUM-relaxation. ii) there is no free unit. That means there is another (non-split) job present on machine i_1, say r. We assign job s to machine i_1 and do not assign job r to machine i_1, while copying all other assignments. We have:

$$v(S_1) \geq \sum_{i=1}^{m} W(i) + w_{s,i_1} - w_{r,i_1}.$$

S_2 is constructed by simply disregarding job s and copying the assignment of all other jobs assigned to machine i_1 in the TUM-relaxation. S_3 is identical to S_2. We arrive at:

$$v(S_2) = v(S_3) = \sum_{i=1}^{m} W(i).$$

Let us now verify whether the claim is true. First, observe that

$$\max\{v(S_1), v(S_2), v(S_3)\} \geq \frac{1}{3}[v(S_1) + v(S_2) + v(S_3)] =$$
$$\frac{1}{3}[3\sum_{i=1}^{m} W(i) + w_{s,i_1} - w_{r,i_1}] = \sum_{i=1}^{m} W(i) + \frac{1}{3}w_{s,i_1} - \frac{1}{3}w_{r,i_1} \geq$$
$$\frac{2}{3}[\sum_{i=1}^{m} W(i) + \frac{1}{2}w_{s,i_1}] = \frac{2}{3}v_{TUM}.$$

Notice that the term $\sum_{i=1}^{m} W(i)$ contains w_{r,i_1}.

Second, since machine i_1 has a free unit with respect two S_2 and S_3, it follows that each fractional machine has a free unit in at least one solution from S_1, S_2, S_3. Thus the claim is valid in case job s is of type 3.

This shows that the claim is true when the solution to the TUM-relaxation features a single split job. Assuming now (by the induction hypothesis) that the claim is true when there are k split jobs, let us proceed to argue that the claim is true for $k+1$ split jobs.

Consider a solution that features $k+1$ split jobs.

Case 1: Suppose that there is a job of type 1, say job s present. Consider now the instance $\mathcal{I}'$ that arises when we delete job s from J and we set $b'_{i_1} = b_{i_1} - 1$, $b'_{i_2} = b_{i_2} - 1$ and $b'_i = b_i$ for all other $i \in M \setminus \{i_1, i_2\}$. Thus, $\mathcal{I}' = (J \setminus s, \{M(j) | j \in J \setminus s\}, b')$. By Lemma 6, an optimal solution to the TUM-relaxation of $\mathcal{I}'$ is given by x_{ji}^{TUM} for $j \in J \setminus s, i \in M$. Hence, this solution features k split jobs and hence the claim is valid for instance $\mathcal{I}'$. Consider machine i_2, and let us assume that machine i_2 is fractional; if not, then the arguments from the $k = 1$ case can be used. From the claim it follows that in one of the three solutions for $\mathcal{I}'$ there is one with a free unit, say S_p, $p \in \{1, 2, 3\}$. So, we schedule job s in S_p on machine i_2, where it takes two units, which is possible by using the free unit. For the other two solutions we schedule job s on machine i_1. This is possible since job s takes only one unit on machine i_1.

Let us now verify whether the claim is true. First, observe that

$$\begin{aligned}\max\{v(S_1(\mathcal{I})), v(S_2(\mathcal{I})), v(S_3(\mathcal{I}))\} \geq \frac{1}{3}[v(S_1(\mathcal{I})) + v(S_2(\mathcal{I})) + v(S_3(\mathcal{I}))] = \\ \frac{1}{3}[v(S_1(\mathcal{I}')) + v(S_2(\mathcal{I}')) + v(S_3(\mathcal{I}')) + 2w_{s,i_1} + w_{s,i_2}] = \\ \frac{2}{3}v_{TUM}(\mathcal{I}') + \frac{2}{3}w_{s,i_1} + \frac{1}{3}w_{s,i_2} = \\ \frac{2}{3}[v_{TUM}(\mathcal{I}') + w_{s,i_1} + \frac{1}{2}w_{s,i_2}] = \frac{2}{3}v_{TUM}(\mathcal{I}).\end{aligned}$$

Second, observe that machine i_1 now has a free unit with respect to S_p and machine i_2 has a free unit with respect to to the other two solutions. It follows that each fractional machine has a free unit in at least one solution from S_1, S_2, S_3. Thus the claim is valid in case job s is of type 1.

Case 2: Suppose that there is a job of type 3, say job s present. Consider now the instance $\mathcal{I}'$ that arises when we delete job s from J and we set $b'_{i_1} = b_{i_1} - 1$, and $b'_i = b_i$ for all $i \neq i_1$. Thus, $\mathcal{I}' = (J \setminus s, \{M(j) | j \in J \setminus s\}, b')$. By Lemma 6, an optimal solution to the TUM-relaxation of $\mathcal{I}'$ is given by x_{ji}^{TUM} for $j \in J \setminus s, i \in M$. Hence, this solution features k split jobs and hence the claim is valid for instance $\mathcal{I}'$. Consider machine i_1, and let us assume that machine i_1 is fractional; if not, then the arguments from the $k = 1$ case can be used. From the claim it follows that in one of the three solutions for $\mathcal{I}'$ there is one with a free unit, say S_p, $p \in \{1, 2, 3\}$. So, we schedule job s in S_p on machine i_1, where it takes two units, which is possible by using the free unit. For the other two solutions we do not schedule job s.

Let us now verify whether the claim is true. First, observe that

$$\begin{aligned}\max\{v(S_1(\mathcal{I})), v(S_2(\mathcal{I})), v(S_3(\mathcal{I}))\} \geq \frac{1}{3}[v(S_1(\mathcal{I})) + v(S_2(\mathcal{I})) + v(S_3(\mathcal{I}))] = \\ \frac{1}{3}[v(S_1(\mathcal{I}')) + v(S_2(\mathcal{I}')) + v(S_3(\mathcal{I}')) + w_{s,i_1}] =\end{aligned}$$

$$\frac{2}{3}v_{TUM}(\mathcal{I}') + \frac{1}{3}w_{s,i_1} =$$
$$\frac{2}{3}[v_{TUM}(\mathcal{I}') + \frac{1}{2}w_{s,i_1}] = \frac{2}{3}v_{TUM}(\mathcal{I}).$$

Second, observe that machine i_1 has a free unit in at least one solution from S_1, S_2, S_3. Thus the claim is valid in case job s is of type 1.

Case 3: Suppose that there is no job of type 1 and no job of type 3. Thus all split jobs are of type 2. Take some split job, say job s. Consider now the instance $\mathcal{I}'$ that arises when we delete job s from J and we set $b'_{i_1} = b_{i_1} - 1$, $b'_{i_2} = b_{i_2} - 1$ and $b'_i = b_i$ for all other $i \in M \setminus \{i_1, i_2\}$. Thus, $\mathcal{I}' = (J \setminus s, \{M(j) | j \in J \setminus s\}, b')$. By Lemma 6, an optimal solution to the TUM-relaxation of $\mathcal{I}'$ is given by x_{ji}^{TUM} for $j \in J \setminus s, i \in M$. Hence, this solution features k split jobs and hence the claim is valid for instance $\mathcal{I}'$. Consider now machines i_1 and i_2; from the claim it follows that in two of the three solutions for these machines, there is a free unit. So, when considering job s, which takes two units on machine i_1 as well as on machine i_2, we schedule it in some solution on machine i_1 and we schedule it in another solution on machine i_2. Since there are two solutions with a free unit for i_1 and i_2, this is always possible. For the remaining solution we simply do not schedule job s.

Let us now verify whether the claim is true. First, observe that

$$\max\{v(S_1(\mathcal{I})), v(S_2(\mathcal{I})), v(S_3(\mathcal{I}))\} \geq \frac{1}{3}[v(S_1(\mathcal{I})) + v(S_2(\mathcal{I})) + v(S_3(\mathcal{I}))] =$$
$$\frac{1}{3}[v(S_1(\mathcal{I}')) + v(S_2(\mathcal{I}')) + v(S_3(\mathcal{I}')) + w_{s,i_1} + w_{s,i_2}] =$$
$$\frac{2}{3}v_{TUM}(\mathcal{I}') + \frac{1}{3}w_{s,i_1} + \frac{1}{3}w_{s,i_2} =$$
$$\frac{2}{3}[v_{TUM}(\mathcal{I}') + \frac{1}{2}w_{s,i_1} + \frac{1}{2}w_{s,i_2}] = \frac{2}{3}v_{TUM}(\mathcal{I}).$$

Second, observe that machine i_1 as well as machine i_2 each have a free unit in at least two solution from S_1, S_2, S_3. Thus the claim is valid in case all split jobs are of type 2.

This completes the proof. □

Although the proof above does not contain an explicit description of an algorithm, it is straightforward to build, given a solution to the TUM-relaxation, the three solutions as described, and to obtain a solution with a weight not less than $\frac{2}{3}$ times v_{TUM}.

The following instance shows that the result is tight, even for P12: we have 3 jobs with $a_1 = a_2 = 1$ and a_3=2, and we have two machines with $b_1 = b_2 = 2$ and $M(j) = M$ for all j. An optimal solution to the relaxation is as follows: $x_{r(1),1} = x_{r(2),2} = x_{r(3),1} = x_{d(3),2} = 1$ with a value of 3. When we follow the

construction from the proof of Theorem 7 we build three solutions such that in each solution exactly two jobs are assigned.

Finally, for completeness we state the following corollary:

Corollary 8 $OPT \leq v_{LP} \leq v_{TUM} \leq \frac{3}{2}OPT$.

Moreover, each of these inequalities can be tight as well as strict as witnessed by the examples throughout the paper.

4 The special case P12: half-integrality and weak persistency

In this section we concentrate on the special case where the resource requirement of a job is independent of the machine. Thus, a job has size 1 or it has size 2 no matter to what machine the job is assigned. In Section 4.1 we show that the LP-formulations M1 and M2 yield half-integral solutions for this special case. Section 4.2 shows the following persistency property for the unweighted version of P12: when a job of size 1 is assigned in the LP-relaxation, it is also assigned in an optimal solution, but not necessarily on the same machine.

We use the following terminology:

- let $J = J_1 \cup J_2$ such that J_1 is the set of jobs of size 1, referred to as the *small* jobs, and J_2 the set of jobs of size 2, referred to as the *large* jobs.
- a vertex or a vector x is called *half-integral* if and only if the value of each of its component x_i is in $\{0, \frac{1}{2}, 1\}$.
- a polyhedron Q is called *half-integral* when each extreme vertex of Q is half-integral.

4.1 Half-integrality of P12

First, we show that for P12 the LP-relaxation and the TUM-relaxation of M2 coincide.

Lemma 9 *For P12 we have* $v_{LP} = v_{TUM}$.

Proof: We show that for P12 a solution to the TUM-relaxation can be modified into a feasible solution to the LP-relaxation with the same weight, implying $v_{TUM} \leq v_{LP}$. Together with Lemma 5 the lemma then follows.

Consider the network constructed in the proof of Theorem 3. In the case of P12, either job j has size 1 which implies that $d(j)$ is not created, or job j has size 2 which implies that $M(j) = M_2(j)$. In the former case, constraints (4) are fulfilled for this job, in the latter case, consider a machine $i \in M_2(j)$ with $x^{TUM}_{r(j),i} + x^{TUM}_{d(j),i} = 1$. We set $x^{LP}_{r(j),i} = x^{LP}_{d(j),i} = \frac{1}{2}$, thereby satisfying constraints (4). Notice that this solution has the same weight. □

This leads to an interesting corollary, namely that LPM2, and by the proof of Lemma 2, LPM1 as well, is half-integral for the special case of P12.

Corollary 10 *LPM1 as well as LPM2 are half-integral.*

Notice further that the proof of Lemma 9 implies that in the LP-relaxation of P12, as constructed as given above, the jobs of size 1 are either assigned to some machine with value 1, or they are not assigned at all. We will use this property in the next section.

Finally, although one could envision that the property of half-integrality holds even for problem P, the following example shows that this is not true. Consider eight machines with capacity two and thirteen jobs $j_1, \ldots, j_{13}$. The first eight jobs $j_i, i = 1, \ldots, 8$ can only be scheduled on machine m_i and have a resource requirement of one. Jobs $j_9, \ldots, j_{12}$ can each be scheduled on two machines; job j_9 has resource requirement two on machine m_1 and one on machine m_2; the jobs $j_{10}, \ldots, j_{12}$ have the same structure on machines m_3 and m_4, m_5 and m_6, and m_7 and m_8, respectively. Finally job j_{13} can be scheduled on m_2, m_4, m_6, and m_8 and has a resource requirement of two on each machine. Figure 2 shows the only way to schedule all jobs in time which leads to splitting job j_{13} into four parts.

Figure 2: Counterexample that shows that half-integrality does not hold for P.

Corollary 11 *For P12: x^{LP} can be found by solving a transportation problem.*

Proof: This follows from Lemma 9 and Theorem 3. □

The following instance shows that there exists an instance of P12 that satisfies the requirements of Theorem 1, i.e., unit weights, exactly two machines per job and machines with capacity 2, for which the value of the relaxation is arbitrarily close to $\frac{3}{2}$ times the value of an optimal solution.

Figure 3: Example of tightness of Theorem 7

Instance. We have an instance with k machines, $k+2$ small jobs and $k-2$ large jobs. We number the small jobs from 1 to $k+2$ and the large jobs from $k+3$ to $2k$. We have $M(j) = \{1, 2\}$ for $j = 1, 2, 3, 4$, $M(j) = \{2, j-2\}$ for $j = 5, \ldots, k+2$, $M(j) = \{j-k, j-k+1\}$ for $j = k+3, \ldots, 2k-1$, and $M(2k) = \{k, 3\}$. An example for $k = 5$ is given in Figure 3. The value of an optimal solution is $k+2$, since machines $3, \ldots, k$ each cannot give more than 1, whereas the value of the relaxation equals $\frac{3}{2} \cdot (k-2) + 4 = \frac{3}{2}k + 1$.

4.2 Persistency

The phenomenon that variables having integral values in a relaxation of the problem attain this same integral value in an optimal solution is called *persistency*. It is a relatively rare phenomenon in integer programming; a well-known

example of it is the vertex-packing polytope (Nemhauser and Trotter [12]; see also Adams et al. [1] and the references therein). We show here that the unweighted version of P12, i.e., with $w_{ji} = 1$ for all $j \in J, i \in M$, satisfies an interesting form of weak persistency. We first prove the following two lemmas.

Lemma 12 *For each optimal solution to the LP-relaxation, there is an optimal solution to P12 such that each job $j \in J_1$ that is assigned in the optimal solution to the LP-relaxation, i.e., for which $x_{ji}^{LP} = 1$ for some $i \in M(j)$, is also assigned in that optimal solution.*

Proof: By contradiction. Suppose that no optimal solution assigns all small jobs that are assigned in x^{LP}. Consider now an optimal solution to P12 that has the following property: it has a maximum number of small jobs on the same machines as in x^{LP}. By assumption, there exists a small job, say job $j \in J_1$, that is assigned in x^{LP} and not present in this optimal solution. Let i be the machine to which j is assigned in x^{LP}. Obviously, machine i has no free capacity in the optimal solution, otherwise we could have improved that solution. Also, at least one of the jobs that are assigned to machine i in the optimal solution, say job j', was not present at that machine in x^{LP}. Thus, replacing j' by j increases the number of small jobs that have the same machine both in the optimal solution and in x^{LP}, thereby violating the property. It follows that there exists an optimal solution that assigns each small job that is assigned in x^{LP}. □

Lemma 13 *For each optimal solution to the LP-relaxation, there is an optimal solution to P12 such that each job $j \in J_1$ that is not assigned in the optimal solution to the LP-relaxation, i.e., for which $x_{ji} = 0$ for each $i \in M(j)$, is also not assigned in that optimal solution.*

Proof: By contradiction. Suppose that in each optimal solution to P12 a nonempty set of small jobs is assigned that is not assigned in x^{LP}. Let S_{OPT} be an optimal solution to P12 with a minimal number of such jobs. Let $j \in J_1$ be a small job that is assigned in S_{OPT}, and is not assigned in x^{LP}. It follows that the $|M(j)|$ machines it is connected to, each have b_i small jobs in x^{LP}, otherwise we can improve the relaxation. Moreover, these $\sum_{i \in M(j)} b_i$ small jobs are connected to machines that each must have b_i small jobs in x^{LP}, otherwise we can improve the relaxation. Proceeding along these lines it follows that we can identify in x^{LP} a set of small jobs, say J', that are connected to a set of machines, say M'. The sets J' and M' have the following properties:

- each machine in M' has b_i small jobs from J',
- all jobs from J' are assigned in x^{LP},
- all connections of jobs in J' are to machines in M', i.e., $\cup_{j \in J'} M(j) = M'$.

Now, consider S_{OPT}. By assumption, in S_{OPT} job j is assigned to a machine in M'. But that implies that some small job from J' that was assigned in x^{LP} has

not been assigned in this optimal solution. Let us now construct an alternative optimal solution to P12. This solution is identical to S_{OPT}, except for each machine $i \in M'$, it uses x^{LP}. Note that this is possible since no job from J' is connected to a machine outside M'. As the constructed solution violates the minimality assumption of S_{OPT}, we arrive at a contradiction. □

Now we are ready to formulate the persistency property that is valid for the unweighted case of P12.

Theorem 14 *For each solution x^{LP} there exists a unique solution x^{OPT}, such that for each $j \in J_1$:*
$x_{ji}^{LP} = 0 \ \forall_{i \in M(j)} \Longleftrightarrow x_{ji}^{OPT} = 0 \ \forall_{i \in M(j)}$.

Proof: This can be shown using the arguments from Lemmas 12 and 13. □

Note that Theorem 14 has the potential to reduce the size of the instances that we need to solve. Indeed, any small job that was not assigned in the relaxation can be discarded from consideration.

Finally, we state the following property of problem P12.

Lemma 15 *There exists an optimal solution to P12 that assigns a maximum number of small jobs.*

Proof: The proof runs along the same lines as the proofs of Lemmas 12 and 13. The idea is to assume that no solution exists in which the maximum number of small jobs is assigned. Then, compare an optimal solution π^* with a solution with a maximum number of small jobs π', and take a small job that is assigned in π' and not in π^*. Then, using the same ideas as used in the proofs of Lemmas 12 and 13, one can show that this job can be assigned in the optimal solution without decreasing the total number of jobs assigned. By doing this iteratively, π^* finally contains the same number of small jobs as π', which contradicts the assumption and proves the lemma. □

5 Conclusion

This paper shows some properties of a generalized assignment problem where the resource requirements are either 1 or 2. We gave two formulations of the problem and showed that the LP-relaxations of the models coincide. Furthermore, we showed that for one of the models there exists a relaxation that leads to an integer program with a totally unimodular constraint matrix, and we show that this relaxation can be solved by a solving transportation problem. This approach leads to a $\frac{2}{3}$-approximation algorithm. For a special case, we showed that the LP-relaxation is half-integral and we derived a weak persistency property.

References

[1] Adams, W.P., J. Bowers Lassiter, and H.D. Sherali (1998), *Persistency in 0-1 polynomial programming*, Mathematics of Operations Research **23**, 359–389.

[2] Aerts, J. (2002), *Random redundant storage for video on demand*, Ph.D. thesis of Eindhoven University of Technology, the Netherlands.

[3] Aerts, J., J. Korst, F. Spieksma (2003), *Approximation of a retrieval problem for parallel disks*, in: Proceedings of the 5th Conference on Algorithms and Complexity (CIAC 2003), Lecture Notes in Computer Science **2653**, 178–188.

[4] Aerts, J., J. Korst, F. Spieksma, W. Verhaegh, and G. Woeginger (2002), *Load balancing in disk arrays: complexity of retrieval problems*, IEEE Transactions on Computers **52**, 1210–1214.

[5] Aerts, J., J. Korst, and W. Verhaegh (2001), *Load balancing for redundant storage strategies: Multiprocessor scheduling with machine eligibility*, Journal of Scheduling **4**, 245–257.

[6] Appa, G. and B. Kotnyek (2002), *A bidirected generalisation of network matrices*, Manuscript.

[7] Chudak, F. and D.S. Hochbaum, (1999), *A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine*, Operations Research Letters **25**, 199–204.

[8] Dawande, M., J. Kalagnanam, P. Keskinocak, F.S. Salman, and R. Ravi (2000), *Approximation algorithms for the multiple knapsack problem with assignment restrictions*, Journal of Combinatorial Optimization **4**, 171–186.

[9] De Farias, I.R. and G.L. Nemhauser (2001), *A family of inequalities for the generalized assignment polytope*, Operations Research Letters **29**, 49–55.

[10] Gemmell, D.J. (1993), *Multimedia network file servers: Multi-channel delay sensitive data retrieval*, Proceedings ACM Multimedia, 243–250.

[11] Hochbaum, D.S. (2002), *Solving integer programs over monotone inequalities in three variables: A framework for half integrality and good approximations*, European Journal of Operational Research **140**, 291–321.

[12] Nemhauser, G.L. and L.E. Trotter, Jr. (1975), *Vertex packings: structural properties and algorithms*, Mathematical Programming **8**, 232–248.

[13] Orlin, J.B. (1993), *A faster strongly polynomial algorithm for the minimum cost flow problem*, Operations Research **41**, 338-350.

[14] Ralphs, T.K. (1993), *On the mixed chinese postman problem*, Operations Research Letters **14**, 123–127.

[15] Ruemmler, C. and J. Wilkes (1994), *An introduction to disk drive modeling*, IEEE Computer **27**, 17–28.

[16] Sanders, P. (2003), *Asynchronous scheduling for redundant disk arrays*, IEEE Transactions on Computers **52**, pp. 1170–1184.

[17] Sanders, P., S. Egner and J. Korst (2003), *Fast concurrent access to parallel disks*, Algoritmica **35**, 21–55.

[18] Savelsbergh, M.W.P. (1997), *A branch-and-price algorithm for the generalized assignment problem*, Operations Research **45**, 831-841.

[19] Shmoys, D.B. and É. Tardos (1993), *An approximation algorithm for the generalized assignment problem*, Mathematical Programming **62**, 461-474.

[20] Vitter, J.S. (2001), *External memory algorithms and data structures: Dealing with massive data*, ACM Computing Surveys **33**, 1–75.